AN A.S. PRATT PUBLICATION
APRIL 2020
VOL. 6 • NO. 3

PRATT'S PRIVACY & CYBERSECURITY LAW REPORT

LexisNexis

Pratt's Privacy & Cybersecurity Law Report

VOLUME 6 NUMBER 3 APRIL 2020

LexisNexis®

QUESTIONS ABOUT THIS PUBLICATION?

For questions about the **Editorial Content** appearing in these volumes or reprint permission, please contact:
Deneil C. Targowski at 908-673-3380
Email: .... Deneil.C.Targowski@lexisnexis.com
For assistance with replacement pages, shipments, billing or other customer service matters, please call:

Customer Services Department at .... (800) 833-9844
Outside the United States and Canada, please call .... (518) 487-3385
Fax Number .... (800) 828-8341
Customer Service Web site .... http://www.lexisnexis.com/custserv/
For information on other Matthew Bender publications, please call

Your account manager or .... (800) 223-1940
Outside the United States and Canada, please call .... (937) 247-0293

ISBN: 978-1-6328-3362-4 (print)
ISBN: 978-1-6328-3363-1 (eBook)

ISSN: 2380-4785 (Print)
ISSN: 2380-4823 (Online)

Cite this publication as:
[author name], [*article title*], [vol. no.] PRATT'S PRIVACY & CYBERSECURITY LAW REPORT [page number] (LexisNexis A.S. Pratt);
Laura Clark Fey and Jeff Johnson, *Shielding Personal Information in eDiscovery*, [6] PRATT'S PRIVACY & CYBERSECURITY LAW REPORT [67] (LexisNexis A.S. Pratt)

An A.S. Pratt™ Publication
Editorial

Editorial Offices
630 Central Ave., New Providence, NJ 07974 (908) 464-6800
201 Mission St., San Francisco, CA 94105-1831 (415) 908-3200
www.lexisnexis.com

MATTHEW BENDER

(2020–Pub. 4939)

Editor-in-Chief, Editor & Board of Editors

Pratt's Privacy & Cybersecurity Law Report is published nine times a year by Matthew Bender & Company, Inc. Periodicals Postage Paid at Washington, D.C., and at additional mailing offices. Copyright 2020 Reed Elsevier Properties SA, used under license by Matthew Bender & Company, Inc. No part of this journal may be reproduced in any form—by microfilm, xerography, or otherwise—or incorporated into any information retrieval system without the written permission of the copyright owner. For customer support, please contact LexisNexis Matthew Bender, 1275 Broadway, Albany, NY 12204 or e-mail Customer.Support@lexisnexis.com. Direct any editorial inquiries and send any material for publication to Steven A. Meyerowitz, Editor-in-Chief, Meyerowitz Communications Inc., 26910 Grand Central Parkway Suite 18R, Floral Park, New York 11005, smeyerowitz@meyerowitzcommunications.com, 646.539.8300. Material for publication is welcomed—articles, decisions, or other items of interest to lawyers and law firms, in-house counsel, government lawyers, senior business executives, and anyone interested in privacy and cybersecurity related issues and legal developments. This publication is designed to be accurate and authoritative, but neither the publisher nor the authors are rendering legal, accounting, or other professional services in this publication. If legal or other expert advice is desired, retain the services of an appropriate professional. The articles and columns reflect only the present considerations and views of the authors and do not necessarily reflect those of the firms or organizations with which they are affiliated, any of the former or present clients of the authors or their firms or organizations, or the editors or publisher.

POSTMASTER: Send address changes to *Pratt's Privacy & Cybersecurity Law Report,* LexisNexis Matthew Bender, 630 Central Ave., New Providence, NJ 07974.

The Seven Layer Cake of Information Security: A Technical Guide for the Non-Technical Reader

By David Kalat[*]

This article offers a primer for understanding key technical concepts and jargon associated with network performance and security.

Lawyers and other professional service providers face a myriad of data security concerns. They are charged with protecting their own data, protecting confidential client data placed in their care, advising clients on how to protect their data, and possibly dealing with or litigating the consequences of a data breach or data loss. Despite these responsibilities, lawyers and information security professionals rarely speak the same language. Both professions are mired in specialized jargon and distinctive ways of looking at the world. This language barrier can inhibit and complicate effective decision-making.

A key principle of information security is the so-called CIA Triad. This is not a reference to a spy agency, but an acronym for Confidentiality, Integrity, and Availability. These three key tentpoles of information systems form the central aim of data security. *Confidentiality* means that if you are not allowed access to some piece of data, you cannot have it. *Integrity* means that data can only be altered by a user who is authorized to modify it. *Availability* is the mirror twin of Confidentiality – it means that if you are allowed access to a piece of data, you can have it. In the event of an incident of data breach or loss, it is critical to quickly establish WHAT went wrong? WHERE did the attack or failure occur? HOW did it happen and how to fix or prevent it?

Information security professionals often make use of a conceptual framework called the Open Systems Interconnection Model ("OSI") to outline the essential components of a computer network. The OSI model offers a general overview of the different functions needed for a computer network to work, and how those functions interrelate. This model is a valuable tool for diagnosing network failures and attacks on the CIA Triad.

By way of analogy, think of a restaurant. There are a variety of different job functions that have to be performed, and generally speaking these are discrete job roles: the host, the bartender, the wait staff, the busboys, the expeditor, the head chef, the prep chef and line cooks, the dishwasher, and so on. In any given restaurant some of these jobs

[*] David Kalat, a director at Berkeley Research Group for Global Investigations + Strategic Intelligence, is a member of the Board of Editors of *Pratt's Privacy & Cybersecurity Law Report*. He is a Certified Fraud Examiner, a Certified Computer Examiner, a Certified Information Systems Security Professional, a Certified Telecommunications Analyst, and a licensed private detective in Illinois and Texas.

may be collapsed together, or subdivided, but an analyst can examine those general job functions to understand how different aspects of the business work together.

The OSI framework describes a series of layers, that are organized in a precise sequence. Each layer consists of a group of protocols for how a particular set of tasks will be performed. These tasks represent essential links in a chain of actions for communicating over a computer network. Consequently, the layers are organized in a precise sequence, with each grouping of protocols performing a necessary function on behalf of the layer immediately adjacent to it.

At one end of the protocol stack is the physical manifestation of electronic data in its most rudimentary form, and the other end of the stack is a human user interacting with the system at its highest level of abstraction. In between are the layers that serve to connect those two worlds.

Network engineers learn the OSI model in order to implement and maintain computer networks. There is value, however, in lawyers and other professional service providers gaining at least a passing familiarity with the model as well – not with the expectation of trying to become engineers themselves, but rather to establish a baseline of technical confidence to help guide informed choices about how to secure data, and what to do in the event of a data breach or loss. This article offers a non-technician's primer for understanding key technical concepts and jargon associated with network performance and security.

LAYER 1: PHYSICAL LAYER

At root, all electronic data has a physical form. Electronic data can be manifested as electrons moving down a wire, pulses of light coursing down fiber optic cables, magnetic charges on spinning metal platters inside a hard disk drive, or other forms of temporarily fixing a representation of electronic information in some kind of media.

The earliest digital computers used vacuum tubes for memory. For example, the Electronic Numerical Integrator and Computer ("ENIAC"), the first general-purpose electronic computer, was originally designed to help calculate artillery firing tables for the Army during World War II. The ENIAC used vacuum tubes like beads on an abacus. Each tube represented either a value between 0 and 9, or a multiple of 10. Vacuum tubes were finicky and fragile, and occupied a lot of space. With an array of 17,468 vacuum tubes, which occupied a space large enough to park a school bus, the ENIAC had an upper capacity of storing just 20 ten-digit numbers.

The ENIAC was a digital computer, but not a binary one – the next generation of computers that followed opted for binary data storage, which was significantly more efficient, and needed far fewer components to represent binary numerical values.

The individual 1s and 0s of binary data are called "bits." The word "bit" is a portmanteau for "binary digit" or "binary unit" and is the smallest, irreducible atom of information. A bit is a single instance of true or false. If a bit is the smallest unit of information, then the smallest, most irreducible thing to *do* with a bit would be to bring two of them together in some kind of interaction to produce an output.

In 1847, mathematician George Boole theorized that all logic could be reduced to a series of true-or-false decision points, and that by doing so, he could express any logical proposition as an algebraic equation. Boole turned human logic into simple math.

Boole achieved this feat by looking closely at the possible outcomes of different true/false scenarios, and realizing there was a limited set of possibilities. He defined each type of operator that would account for the different combinations. For example, the Boolean operator "AND" returns a value of True only if both of the input values are True. The "OR" operator however returns a True value if any one of the inputs is True. The "Exclusive OR" ("XOR") operator only returns True if one and only one of the inputs is True. "NOT" behaves like a child's game of Opposite Day, and simply negates the input.

Input X	**Input Y**	**AND**	**OR**	**XOR**	**Input X**	**NOT (-x)**
0	0	0	0	0	0	1
1	0	0	1	1	1	0
0	1	0	1	1		
1	1	1	1	0		

For many lawyers, the term "Boolean operators" has entered the lexicon in the context of formulating search queries in Google or Relativity, but that is just one specialized application of Boolean logic. The power of Boolean operators is that it allows problems to be expressed as a series of True/False decision trees, which can then be represented using the components of Boole's symbolic logic.

In the late 1800s and early 1900s, telephone companies started building vast networks of electrical relays that took in electrical inputs, and then either opened or closed the relay based on those inputs. Curiously, no one had quite recognized the significance, until the 1930s when an inventor named Claude Shannon pointed out that the telephone network was a massive network of Boolean switches capable of executing any logical operation. That momentous fusion of telecommunications and mathematics gave birth to computer science, and ushered in the Information Age.

The physical layer is especially relevant in digital forensics. All persistently stored electronic data has a physical manifestation where the individual 1s and 0s are represented. Forensic examiners will often access the physical layer to access that persistently stored information, without having to actually turn on the machine or boot up its operating system. There are circumstances where residual information remains encoded by the physical condition of storage media even though the logical operation of the system no longer recognizes it as active data. Put another way, deleted data can persist and be recovered in whole or in part from a computer by extracting it from the physical layer even if the upper layers of the system think it has been destroyed.

LAYER 2: DATA LINK LAYER

The phrase "Information Superhighway" was popularized in the 1990s as shorthand for the internet. The phrase has since fallen out of favor, but its metaphor is useful for understanding the Data Link Layer, where interconnected computer systems communicate locally. If the internet is the Information Superhighway, then there must be Information Streets and Back Roads to allow data to move from its local residence up onto the wide network highways. These local passageways are the Local Area Network ("LAN").

The various networked components of a LAN form a single broadcast domain, and are usually connected by either a wired Ethernet connection or wireless WiFi. These components can consist of computers, tablets, and phones as well as single-purpose devices such as routers, printers, and hubs. The fact that these components coexist on the same broadcast domain means that the data transmitted within the LAN passes through each of the devices on that circuit.

This is one of the counterintuitive consequences of dealing with binary electronic data, and so it is worth the effort to fully understand the implications. To that end, another metaphor may be helpful. Imagine a postal worker, attempting to deliver a letter to someone. In the *physical* world, different types of matter clearly separate and distinguish the postal delivery person, the letter, the mailbox, the space through which the letter must be moved to get it into the mailbox, and so on. The act of delivering a letter involves moving a distinct physical object from one position in three-dimensional space to another. Inside the world of electronic data, however, these pieces are all made of the same stuff, signals and pulses that represent ones and zeroes. The letter is just a pattern of signals that travels along a wave, and the act of delivering that letter involves sending that wave through all of the components on that circuit.

In practical terms, then, transmitting an electronic message from one device to another on a LAN means transmitting it through multiple devices but in such a way that only the intended recipient reacts to it. To accomplish this, data is transmitted within a LAN inside what are called *frames*. Frames are like envelopes. They wrap around the data with identifying information about the address of the intended recipient. The various stations in that local network ignore the frames that are not addressed to them. The target machine with that address unpacks the data out of the frame to respond accordingly.

From a security perspective, there are two key takeaways.

The first is, because data passes through other stations on a local network that are not the sender or recipient, any data that is not encrypted is exposed to the risk of eavesdropping. In fact, one could quibble whether it would even count as "eavesdropping" to read unencrypted traffic that was passed openly on a shared network. Continuing with the post office metaphor, the only way to read the contents of a sealed envelope is to break open the envelope. That scenario would be akin to using encryption. Messages sent without encryption more closely resemble a post card – every time that postcard passes through a node of the network, that node can read what is on it.

This highlights the security risk of using public WiFi, or of failing to secure a private LAN with the appropriate encrypted controls. A user logged on to a public WiFi network is potentially sharing all unencrypted communications with anyone or everyone else in that network. Similarly, enabling WPA2 encryption with a good, strong password is critical. It is important to maintain control over who has access to a shared network, and sensitive data and communications can be protected using encryption to limit what any eavesdropper within that network can intercept.

The second takeaway is, if a malicious intruder wants to target a given computer, to put malware on it or to extract data from it – they have to get *to* it first. To do so means getting that computer's Media Access Control ("MAC") address, in order to correctly address data link frames through the correct local area network to that specific recipient.

Every networked device has a unique MAC address that is assigned by the hardware manufacturer. A MAC address is like the Social Security Number for a piece of networked computer hardware – it is meant to be a unique identifier, and if someone were to steal it they could masquerade as that device. Without the right MAC address, the frames pass through the target computer unopened. Only if the attacker has your MAC address can they even interact with your computer. These addresses are not published publicly and are not viewable from outside the LAN – they are essentially secret addresses shared with trusted friends.

In terms of data security, this means that an external attacker cannot target your computer unless they have a way of getting your MAC address. There are technological attacks on local area networks that *can* be used, but by far and away the easiest way for an attacker to get to your machine is to trick you into voluntarily accepting that connection. For example, a "phishing" email is designed to appear sufficiently convincing to entice the recipient into clicking on a link or opening an attachment. That action essentially instructs the user's computer to request the malicious data, and to disclose its MAC address to receive it. This opens up a door that otherwise is hard for the attacker to get past.

Technological protections on networks can only go so far. In the end, any user of a networked computer is their own last and most important line of defense.

LAYER 3: NETWORK LAYER

The distinction between the internet and a Local Area Network is *routing*. On a LAN, every station is interconnected and shares data. On the internet, data will pass through a series of nodes – think of them as data intersections – where routing decisions will have to be made about which direction to take next towards the destination. Many computers may be involved in the sequence of hops from one end to the other, but the data does not pass through every computer in the world.

At the Network layer, instead of identifying the intended destination by MAC address, the addressing scheme is the "Internet Protocol" or "IP." Unlike a MAC address, an IP address is not a unique identifier permanently given to a machine at the time of manufacture, but rather an address (temporarily) assigned by a central registry.

It would be useful if every IP address were a unique identifier for a machine on a network – and if all systems upgraded to the newer IPv6 standard there would be enough IP addresses to accommodate that goal. Most systems in use currently, however, use the older IPv4 standard, which does not have enough unique numbers for all the networked devices that now exist.

As a consequence of this limitation, IP addresses get parceled out into two categories. One set are unique, never-reused "public IP" addresses that definitively identify a specific machine, but due the limited number of available public IPv4 numbers their use is limited to top-level systems. A second set of "private IP" addresses are not unique, but are only used at a local level within a given LAN.

Most individual users' computers are identified on their local networks by one of a limited supply of IP addresses that are reused for such private addresses. When such a user gets onto the internet and accesses a web page, the remote web server will see incoming traffic from the unique public IP address of the router that handles that user's network – and if multiple users on the same network all access the same web page, each of those users would appear to the web server as coming from the same public IP address. Only the LAN's router would know the individual private IP addresses for the different machines. Consequently, IP information can often be useful in localizing where a connection originates geographically, but rarely is sufficient by itself to identify a specific user.

IP addresses do however identify different Local Area Networks, and can be used to direct traffic between them. Transmitting data between IP addresses means dividing the data into small "packets" and wrapping the packets in metadata that identify the IP addresses of the sender and destination. At each node, the packet will encounter a router that reads that IP address information and decides what is, at that very moment, the most efficient route to that destination. That decision will not necessarily be the same route for every packet for the same data set.

LAYER 4: TRANSPORT LAYER

It is at the Transport Layer that the data is carved into the packets that get sent out in the Network Layer. When a client and server system establish a connection, those systems first engage in a handshake process to select which protocol will be used to packetize the data. This initial protocol selection is negotiated so that *before* user data is exchanged, protocols are in place to secure the integrity of that data.

For example, an email server will listen on an open port until it hears a connection request from an email client such as Outlook. The two pieces of software, client and server, go through a specific process to authenticate each other and select an encryption algorithm prior to exchanging your data.

Two principal standards for packetizing data at the Transport layer are in general use. Transmission Control Protocol, or "TCP," is used for emails, web pages, and other types of data and communications that are not time-sensitive. TCP is said to be "reliable," which simply means the protocol has an error-checking and error-correcting faculty, so that if a packet gets lost or corrupted, it will get fixed or re-sent so that the recipient side gets the complete correct data. That approach is unhelpful for situations where the data needs to remain in a continuous, uninterrupted stream, such as streaming, or phone calls that use Voice Over IP ("VOIP") transmission. User Datagram Protocol ("UDP") maintains the constant flow of data by attempting to repair missing or damaged data on the fly without waiting for replacements. Anyone who has been on a VOIP call in which the speaker's voice became garbled or started to skip has experienced the UDP transport protocol trying to make do around some missing packets.

How does software identify and repair damaged data on the fly? By adding redundancy in carefully selected ways. As an analogy, when a reader encounters a typographical error in a written document, it is often possible to recognize what the misspelled word was intended to be, thanks to the information conveyed by the *other* letters in the misspelled word, and the context of the other words around it. That context places limits around what the misspelled letter or letters could be – usually not many possible choices make any sense.

Error checking and correcting codes in binary data work much the same way. Extra bits are added in specific locations, and their values are chosen based on the values of specific selected bits in other locations. If errors or interference cause certain bits to get flipped, the error checking sequences become mathematically nonsensical in ways that can only be resolved if the corrupted bit or bits get turned back to their correct values.

LAYER 5: SESSION LAYER

The Session Layer contains the protocols responsible for opening, managing, and ultimately closing the connections between the client and server. Many of these protocols revolve around login credentials and passwords.

In many ways it is surprising that passwords are still used to secure data, considering how profoundly insecure they are and have always been. The first recorded use of passwords for a computer system was in 1961. This was also, it must be said, the first recorded instance of password theft.

MIT's Compatible Time-Sharing System, or "CTSS," was a pioneering computer science research project. The CTSS was intended to provide a working environment for multiple users who would share access. Each of those users had their own domain of research and their own files. Somehow the CTSS needed to be able to distinguish between its many users, to present each one with only the materials authorized to them.

One of the CTSS researchers was Ph.D. candidate Allan Scherr. Like everyone else working on the system, he was assigned four hours per week of computer time. Scherr figured out a sneaky way to get more than four hours. He discovered that it was possible to submit a print request *for any file*, and that there was nothing preventing him from submitting a request to print out the master password file. One weekend in the spring of 1966, Scherr obtained a complete list of everyone else's passwords, becoming the first computer password hacker. In an especially clever flourish, he then gave copies to a few other users, to make it harder for unauthorized access to be traced back to him alone. Indeed, until Scherr confessed at the 25th anniversary of CTSS, no one suspected him.

Scherr's hack illustrates a central problem with managing passwords on a network. That master password file represents a single point of failure – all an attacker needs is to get that one file, to expose all of the users' logins.

So if storing passwords in a master file is a bad idea, how do you administer them safely?

In the 1970s, a new approach to password security involved not storing them at all – at least not in a recognizable form. Instead of storing the passwords in a file vulnerable to being leaked or stolen, each password was *hashed* first. The term "hash" comes from cooking – the practice of transforming something by slicing and dicing it. A mathematical hash takes a chunk of electronic data – such as a password – and processes it through a complex algorithm to reduce it into an encrypted slug of data. This transformation is strictly one-way. There is no way to "un-hash" that encrypted slug back into the original password, any more than a cook can turn hash browns back into a raw potato.

Hashes have many useful functions in computer science, but in terms of password security the value proposition is a way to store passwords in a form supposedly impervious to attack. When a user attempts to log-on and enters their password into the system, the interface hashes that input and compares it against the stored list of password hashes. If the two hashes match, the user is authenticated into the system. If that list of password hashes were ever to be leaked or stolen, no one could reverse them back into the users' passwords.

At least, that was the idea. The problem was not the hashing algorithm, which was impervious to direct attack. Instead, the problem was human nature. There are just basic fatal flaws in how users choose their passwords – routine human habits that make for catastrophic security lapses.

For one thing, most users choose the same password. A small selection of possible passwords ("password," "qwerty," "1234") account for the vast majority of the passwords in use at any given time. Consequently, these duplicate passwords share duplicate hashes. An attacker could grab the central password hash file, containing a list of absolutely uncrackable hashes, and see that a quarter of them were exactly the same, and be able to deduce those users were all using something like "1234" as their password.

The solution, pioneered in 1979, is to "salt the hash." This involves adding random data into the systems that process user passwords. Using this technique, a hundred users who all used the same password would end up with a hundred different hashes, each distinguished by a unique piece of random "salt." This system of "salted hashes" remains in use today, 40 years after it was first published, and continues to be recognized as the gold standard of password security systems. Even so, it is only as secure as the conscientious security-mindedness of the user.

LAYER 6: PRESENTATION LAYER

The second to the last layer provides the intermediation between human language and machine language. In this context, "machine language" does not refer to software programing languages – software code can have an odd syntax but are readable by humans. Instead, "machine readable" here means binary.

To illustrate the issues involved in mapping human language onto binary data, consider this sentence: "Hi Sue, do you want to play trivia on Monday?" Accounting for all the letters, spaces between words, and punctuation, that statement requires 45 characters. How many bits are needed to encode and transmit this 45-character message?

Since each bit has two possible values, 1 or 0, each additional bit doubles how many values the bit sequence can hold. For a sequence of 8 bits, because each bit has two values, the total set of possible combinations is 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2. One 8-bit byte has 256 possible values. That is more than enough different values to represent all 26 letters in the English alphabet in both upper and lowercase, the digits 0-9, all punctuation marks in general usage, and a number of specialized letters borrowed from European languages. There are not enough values to represent all the characters needed for Russian, Chinese, Japanese, and so on, but for anything in English, it is possible to use one byte per character.

The American Standard Code for Information Interchange ("ASCII") table is a well-established standard for mapping the 256 possible binary values of 8-bit bytes onto a character set of predominantly English orientation. Each byte value has a corresponding character. At the very simplest level, a Presentation Layer service converts the characters of (English) text into 8-bit bytes.

Presentation Layer services can do much more than that, though. For example, instead of directly mapping text onto bytes, compression is a tool for mapping text onto a smaller set of binary data that stores the same information more compactly.

The same techniques are widely used in the physical world, and offer a helpful analogy. Imagine that there is a confidentiality agreement that needs to be signed individually by 20 different people. Instead of saving 20 complete copies of the agreement, each of which is identical save for the signature page, it would be more efficient to save a single copy and 20 individual copies of the signature page alone. Compression algorithms execute the same concept on electronic data, but identifying redundant sections that only need to be saved once.

Encryption, however, is a technique for saving information in a transformed state intended to obscure its meaning. Compression and encryption are actually closely related and share a lot of technological characteristics. The key difference is that the purpose of compression is to easily change the converted data back into the original information, whereas the purpose of encryption is to restrict the ability to turn the altered data back to the original information to just authorized parties.

Electronic encryption achieves its robust strength by leveraging a specific Boolean operator with special properties.

Cryptography involves taking a message to be encrypted (the "plain text") and performing a transformative operation on it to produce an encrypted output (the "cipher text"). In order for an authorized recipient of the cipher text to be able to restore the plain text message, there must be a way to reverse that cryptographic operation. This is just another way of saying that cryptography means combining a plain text input with a second input that serves as a key, and that operating that same key on the cipher text restores the original plain text input. On a bit-by-bit level, cryptography is a Boolean operation.

For an unauthorized third party to successfully decrypt the cipher text back into the plain text, they have to undo the operation that encrypted it – which is tantamount to working out the secret key. To have secure encryption, it is necessary to make the reverse operation impossible for anyone who does not already have the secret key.

Language is rich with underlying patterns, and as a result it is very difficult to hide these patterns. The most common letter in the English language is e; the most common first letter of a word is a; the most common word is the. These and other observations allow statistical analysis to deduce what the original characters were. Puzzle fans entertain themselves by working these sorts of schemes out as a leisure time activity. In order to defeat these kinds of attacks and remove any of the underlying patterns inherent in the original data, a strong encryption scheme needs to be both unpredictable and non-repeating. An encryption scheme that results in an unpredictable and non-repeating cipher text is secure from statistical attack.

Interestingly, this means that the security of an unpredictable and non-repeating encryption scheme does not depend on the secrecy of the algorithm, and indeed no longer benefits from that secrecy at all. Securing the decryption tools does not mean keeping the encryption method itself secret. In fact, the opposite is true. Information security professionals advocate for making the inner workings of an encryption scheme public. This is known as Kerckhoff's Principle: the security of the system does not depend on secrecy. All of the most commonly used strong encryption schemes are in fact publicly known algorithms. The security community benefits from this public knowledge by ensuring that algorithms are rigorously tested and evaluated. Secret algorithms however tend to hide their flaws until it is too late.

Think of it like an ordinary house lock – there is no great secret to how deadbolt locks work, but you need the right key to turn it. In the world of electronics, the key is not a physical lump of metal hanging off your keychain, it is a piece of data. Securing encrypted data therefore does not mean hiding how it was encrypted, but in hiding the key.

Now consider the Boolean operators used in information technology:

Input X	**Input Y**	**AND**	**OR**	**XOR**	**Input X**	**NOT (-x)**
0	0	0	0	0	0	1
1	0	0	1	1	1	0
0	1	0	1	1		
1	1	1	1	0		

The operator called Exclusive OR ("XOR") is the foundation of strong encryption. That is because it exhibits an ambiguous relationship between output and input that the other operators do not. Conversely, the AND and OR operators each have a giveaway "tell" that makes reversing any operations dependent on them too easy.

Operations involving AND can only produce a 1 if both inputs are 1s. Consequently, any attacker trying to decrypt a string of binary digits created using AND can immediately know the inputs that led to every 1 in that string. Similarly, the OR operator can only produce a 0 if both inputs are 0. An attacker could immediately reverse all the 0s in the cipher text back to their original inputs. The remaining bits would still need to be brute-forced, but a significant leap forward would have been made towards working out the original plaintext.

But then there's XOR. For every possible output value, 1 or 0, there are two possible inputs. For every bit in the cipher text, there are no hints.

As previously noted, a typical email is 75 Kilobytes. Because there are 1024 bytes in a kilobyte, a 75 Kb email is 600,000 bits. Every one of those 600,000 bits, and every one of those has two possible decryption operations. In other words, two times two *600,000* times.

Furthermore, encryption algorithms do not just do a single pass through one XOR – there are multiple rounds, the output of one being reprocessed and sent through again, stacking XOR transformations on top of each other. Each pass exponentially expands the number of decision points that have to be brute forced.

There is a famous Hindu legend about a king and a travelling wise man who challenged the king to a game of chess. Before the game started, the king asked the wise man what he wanted if he won the game. The man simply asked for rice.

The king was puzzled – *really? You don't want gold, or silk, or land or anything? Just rice?* The wise man said, "Give me one grain of rice for the first square on the chess-board. Two grains on the second square. Four on the third, and so on, doubling each time, to fill the board."

The king agreed.

They played the game, and the wise man won. The king was true to his word, and started counting out rice to pay his debt. 1 grain, 2 grains, 4 grains . . . on the eighth square, 128 grains. He started on the second row – 256 grains, 512 grains, 1024 . . . and it started to dawn on him. It sounded so simple at first, but doubling is surprisingly powerful. On the fourth row of the chessboard the king needed 2.1 billion grains. By the final square, he would need to put down 9 followed by 18 zeroes grains of rice. This was more than his vast wealth could afford.

In the legend, the wise man reveals himself to be the god Krishna. He agreed the kingdom could pay him over time, and pilgrims still go to the temple to enjoy a rice pudding in honor of the king's debt. We can simply note that every pass through XOR is pulling Krishna's trick.

LAYER 7: APPLICATION LAYER

The topmost layer in the OSI Model is the interface between the human user and the computer system. The Application Layer is where data is consumed. This is where an attacker would focus their energies to get access to bank accounts, personal information, critical business records, and all the live active data that users typically associate with electronic data.

By way of illustrating for the Application Layer and the preceding layers of the OSI model interconnect to join the human experience to the world of electronic bits, consider what happens when a user navigates to a web page using a browser.

A website is a set of software code running on a server, and contains various pieces of data. When a user visits that site, what actually happens is that the web browser application reaches out to the web site and asks for that data to be sent to it. In other words, the human user at the human-machine interface interacts at the Application Layer, creating the message "*hey I'd like to see this web page please*."

The Presentation Layer converts that message into the machine-readable code, passes it through a Session Layer connection, and uses Transport Layer protocols to send it across the internet. In this case, to see a web page, the Transport Control Protocol with guaranteed error correcting and delivery is used to communicate between nodes identified by IP addresses. The Network Layer communicates packets across the network of internet-enabled computers and devices to get from the user's computer to the web server. Along the way, the Local Area Network maps the outgoing request onto the MAC address of the originating computer so that when the web server sends its response back, the response will be delivered to the correct computer in the correct browser and everything else in that LAN will ignore that data. The individual bits, a wave of electrons, reaches the web server. The server responds by sending back the bits that constitute the web page in question, which are sent as packets, passed through the internet with headers identifying the IP address of the router used by the originating computer. The router maps that incoming data onto the correct MAC address, and the packets are placed in frames to be delivered to the browser. *Ta da.*

One of the most common forms of Application Layer attacks is to send an unreasonable volume of requests to the same web server at the same time. The server will dutifully try to respond to each one, but will get overloaded and lock up, unable to comply with the incoming requests. This is called a Distributed Denial of Service attack, and is a very common way of temporarily bringing down a web site.

Intriguingly, that common form of Application Layer attack depends on having previously accomplished another type of Application Layer attack. Because the technical limitations that will cause a web server to freeze up if it tries to respond to too many requests all at once also prevent any one computer from being the author of all those requests. In order to cause a Distributed Denial of Service attack, an attacker will need a number of different computers controlled all at once.

The first step in enslaving multiple computers into such a zombie army entails pushing malicious software, malware, to those computers. As previously discussed, in order to get data to a specific machine there needs to be a connection between MAC addresses, which are not publicly disclosed. Therefore, the easiest way to get malware onto a computer is to trick the human user into directing their computer to *ask* for the malware. This is often done through a phishing email, something that looks convincing enough to entice the recipient into clicking on a link or opening an attachment. This action serves to instruct the computer to request the data, and to announce its address to facilitate the connection. Tricking a user into responding to a phishing request effectively bypasses all other technological safety controls, because the nature of the system is to facilitate what the users want to do.

Along those same lines, there is another way for an attacker to commandeer a zombie army to direct a Distributed Denial of Service attack, that also depends on the lack of security mindedness of users. So-called "smart" appliances and other "Internet of Things" devices are internet-enabled devices, but ones which users often discount when thinking about data security. Users are more likely to be concerned with security when dealing with sensitive data, such as logging into bank accounts, making a purchase, or sending confidential data. The same level of awareness is rarely invoked when dealing with a Roomba. *What's a bad guy gonna do? Hack into your home and clean the floors?*

Smart appliances often come with factory-installed default passwords, which users may not think to change. Attackers can then rely on the known default passwords to remotely connect to large numbers of internet enabled appliances, turning them into zombified weapons to conduct other attacks.

That's what happened on October 21, 2016. An attacker used default passwords to access tens of millions of internet enabled smart devices and then used them to temporarily shut down some of the most marquee names on the web: Twitter, Paypal, Spotify, Mashable, CNN, *The New York Times*, *The Wall Street Journal*, and Yelp.

CONCLUSION

The vast majority of data breaches are the result of the errors, oversights, or deliberate mischief of individual employees. A 2015 study by CompTIA attributed 52 percent of all incidents to "human error." Statistics reported by the UK's Information Commissioner's Office in 2016 put the figure at 62 percent. Michael Bruemmer, vice president of Experian Data Breach Resolution, noted that "about 80% of all the breaches we service have a root cause in some type of employee negligence." Instead of instituting a campaign touting "Only you can prevent data breaches," both the victim organizations and the press usually foster a narrative of data breaches that emphasizes the frightening technological offense, rather than the humble human defense.

The overall impression is that data breaches are the result of enemy nation states, hacktivists, organized crime, and other evil forces who have leveraged the skills of armies of young technologists to unleash unrelenting attacks on American businesses and their information assets – which is of course true. The presence of barbarians at the gate should be the reason to ensure the day-to-day users who operate the gate do not unwittingly them in.

Gartner has estimated that cybersecurity spending will exceed $1 trillion over the next five years – to be spent in large measure on technology and cybersecurity professionals.

Meanwhile spending on cybersecurity employee awareness training is estimated to reach just $10 billion over the next 10 years – a far smaller slice of the pie. Yet a single employee's mistake can open the gate to the barbarians and render all that supporting technology moot.

Where the responsibility for information security once was the province of computer science specialists, the PC revolution shifted that responsibility onto individual users who sometimes struggle to know how to turn their computers on. Yet it is the individual users who form the ultimate boundary between an organization's information system and the outside world, and their behavior has direct and tangible effects on the security of that boundary.

Nowhere is this more evident than in the case of phishing attacks. No matter how well hardened a network, or how sophisticated and robust the technology girding its defense, the most vulnerable spot will always be the channels of communication that necessarily flow into the network from the outside.

Not clicking on links in emails, not re-using passwords, deploying available tools like encryption and two-factor authentication, and running regular backups are relatively easy and inexpensive actions that, if they were simply more common and reflexive on the part of users, would have imposed significant barriers to the attackers in many of the most headline-grabbing recent incidents.

Effective security awareness training can be the differentiator between those organizations that are merely targeted by attackers, and those that are actually victimized by them. The old story goes that two men in the jungle spot a lion. One man starts lacing up his running shoes. The other asks, "Do you really think you can outrun a lion?" "No," the runner replies, "I think I can outrun you." There are determined attackers out there, with deep resources to launch coordinated, targeted, sophisticated attacks. Why make it easy for them?